module *MProlog/Terms*
  imports *INT*
  imports *QID*
  . :→ *GroundTermList*
  _,_ : *GroundTermList GroundTermList* → *GroundTermList* *associative identity*(.) *precedence:* 20
  _,_ : *TermList TermList* → *TermList* *associative identity*(.) *precedence:* 20
  p : *Qid* → *Atom*
  v : *Qid* → *Var*
end

module *MProlog/PreSyntax*
  imports *MProlog/Terms*
  ; :→ *RuleList*
  _. : *Term* → *Rule* *precedence:* 30
  _:-_. : *Term TermList* → *Rule* *precedence:* 30
  __ : *RuleList RuleList* → *RuleList* *associative identity*(;) *precedence:* 40
  _(_) : *Atom GroundTermList* → *GroundTerm* *precedence:* 10
  _(_) : *Atom TermList* → *Term* *precedence:* 10
  grouped : *TermList* → *Term*
  undef :→ [*RuleList*]
  $\dfrac{F\ .}{F\ :-\ .\ .}$
end

module *MProlog/GroupedRules*
  imports *MProlog/PreSyntax*
  __ : *GroupedRulelistSet GroupedRulelistSet* → *GroupedRulelistSet* *associative commutative identity*(*noORL*)
  {_,_} : *Atom RuleList* → *GroupedRulelistSetElement*
  noORL :→ *GroupedRulelistSet*
end

module *MProlog/PreprocessToGrouped*
  imports *MProlog/GroupedRules*
  groupIt : *RuleList GroupedRulelistSet* → *GroupedRulelistSet*

$groupIt((\underline{\ A\ (\ Tl\ )\ :\text{-}\ Fl\ .}\rangle, \langle\ \{\ A\ , \langle\underline{\ \cdot\ }) \}\rangle)$
$\cdot \qquad\qquad\qquad\qquad\qquad\qquad A\ (\ Tl\ )\ :\text{-}\ Fl\ .$

$groupIt((\underline{\ A\ (\ Tl\ )\ :\text{-}\ Fl\ .}\rangle, \langle\underline{\ \cdot\ }\rangle)) \qquad otherwise$
$\cdot \qquad\qquad\qquad\qquad\qquad\qquad \{\ A\ ,\ \ A\ (\ Tl\ )\ :\text{-}\ Fl\ .\ \}$

$\underline{groupIt(;, srs)}$
srs

end

module *MProlog/Syntax*
imports *MProlog/GroupedRules*
end

module *MProlog/Builtins*
imports *MProlog/Syntax*
! :⟶ *BuiltInTerm*
arithmeticError :⟶ [*TermList,FindResult*]
eq-pr :⟶ *GroupedRulelistSet*
eqi :⟶ *Atom*
eqi-pr :⟶ *GroupedRulelistSet*
equal :⟶ *Atom*
eval : *GroundTerm* ⟶ *Int*
fail :⟶ *BuiltInTerm*
gt :⟶ *Atom*
gt-pr :⟶ *GroupedRulelistSet*
gte :⟶ *Atom*
gte-pr :⟶ *GroupedRulelistSet*
i :⟶ *Atom*
intOrNat : *Int* ⟶ *GroundTerm*
is :⟶ *Atom*
listCons :⟶ *Atom*
listNil :⟶ *Atom*
lt :⟶ *Atom*
lt-pr :⟶ *GroupedRulelistSet*
lte :⟶ *Atom*
lte-pr :⟶ *GroupedRulelistSet*
member :⟶ *Atom*

$member\text{-}pr :\longrightarrow GroupedRulelistSet$
$minus :\longrightarrow Atom$
$n :\longrightarrow Atom$
$not :\longrightarrow Atom$
$not\text{-}pr :\longrightarrow GroupedRulelistSet$
$plus :\longrightarrow Atom$
$prelude :\longrightarrow GroupedRulelistSet$
$quotient :\longrightarrow Atom$
$times :\longrightarrow Atom$

$$\frac{}{\{\ equal\ ,\quad equal\ (\quad v(\ 'X)\ ,\ v(\ 'X)\ )\ .\ \}}\ eq\text{-}pr$$

$$\frac{}{\{\ eqi\ ,\quad eqi\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ :\text{-}\quad lte\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ ,\quad gte\ (\quad v(\ 'T')\ ,\ v(\ 'T)\ )\ .\ \}}\ eqi\text{-}pr$$

$$\frac{}{\{\ gt\ ,\quad gt\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ :\text{-}\quad is\ (\quad n\ (\ v(\ 'X)\ )\ ,\quad minus\ (\quad minus\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ ,\quad n\ (\quad s\ 0\ )\ )\ )\ .\ \}}\ gt\text{-}pr$$

$$\frac{}{\{\ gte\ ,\quad gte\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ :\text{-}\quad is\ (\quad n\ (\ v(\ 'X)\ )\ ,\quad minus\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ )\ .\ \}}\ gte\text{-}pr$$

$$\frac{}{\{\ lt\ ,\quad lt\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ :\text{-}\quad is\ (\quad n\ (\ v(\ 'X)\ )\ ,\quad minus\ (\quad minus\ (\quad v(\ 'T')\ ,\ v(\ 'T)\ )\ ,\quad n\ (\quad s\ 0\ )\ )\ )\ .\ \}}\ lt\text{-}pr$$

$$\frac{}{\{\ lte\ ,\quad lte\ (\quad v(\ 'T)\ ,\ v(\ 'T')\ )\ :\text{-}\quad is\ (\quad n\ (\ v(\ 'X)\ )\ ,\quad minus\ (\quad v(\ 'T')\ ,\ v(\ 'T)\ )\ )\ .\ \}}\ lte\text{-}pr$$

$$\frac{}{\{\ member\ ,\quad member\ (\quad v(\ 'H)\ ,\quad listCons\ (\quad v(\ 'H)\ ,\ v(\ 'T)\ )\ )\ .\quad member\ (\quad v(\ 'H)\ ,\quad listCons\ (\quad v(\ 'H')\ ,\ v(\ 'T)\ )\ )\ :\text{-}\quad member\ (\quad v(\ 'H)\ ,\ v(\ 'T)\ )\ .\ \}}\ member\text{-}pr$$

$$\frac{}{\{\ not\ ,\quad not\ (\ v(\ 'T)\ )\ :\text{-}\quad v(\ 'T)\ ,\ !\ ,\ fail\ .\quad not\ (\ v(\ 'T)\ )\ .\ \}}\ not\text{-}pr$$

$$\frac{}{eq\text{-}pr\quad eqi\text{-}pr\quad gt\text{-}pr\quad gte\text{-}pr\quad lt\text{-}pr\quad lte\text{-}pr\quad member\text{-}pr\ not\text{-}pr}\ prelude$$

$$\frac{eval(\_)}{arithmeticError}\quad otherwise$$

$$\frac{eval(\ i\ (\ I\ ))}{I}$$

$$\frac{eval(\ minus\ (\quad T\ ,\ T'\ ))}{eval(\ T)\ \text{-}\ eval(\ T')}$$

$$\frac{eval(\ n\ (\ N\ ))}{N}$$

$\dfrac{eval(\ plus\ (\ \ T\ ,\ T'\ ))}{eval(T)\ +\ eval(T')}$

$\dfrac{eval(\ quotient\ (\ \ T\ ,\ T'\ ))}{eval(T)\ quo\ eval(T')}$

$\dfrac{eval(\ times\ (\ \ T\ ,\ T'\ ))}{eval(T)\ *\ eval(T')}$

$\dfrac{intOrNat(I)}{i\ (\ I\ )}$ *otherwise*

$\dfrac{intOrNat(N)}{n\ (\ N\ )}$

$\dfrac{intOrNat(arithmeticError)}{fail}$

end

module *MProlog/K*

imports *MProlog/Syntax*

$\_\curvearrowright\_ : K\ K \rightarrow K$ *associative identity*(.)

end

module *MProlog/VarAnalysis*

imports *MProlog/Syntax*

$\_\#\_ : VarSet\ VarSet \rightarrow VarSet$ *associative commutative identity*(mt)

$\_in\_ : Var\ VarSet \rightarrow Bool$

$mt :\rightarrow VarSet$

$vars : TermList \rightarrow VarSet$

$X\ \#\ \underline{X}$

$vars(\underline{\ T\ ,\ \ T'\ ,\ Tl}) \# \dfrac{\cdot}{vars(T)\ \#\ vars(Tl)}$ $\quad \underline{T'}$

$vars(\underline{\ \_\ (\ Tl\ )}) \quad Tl$

$vars(\underline{grouped(Tl)}) \quad Tl$

$\dfrac{X\ in\ \langle X \rangle}{true}$

$\frac{\_ \; in \; \_}{false}$ *otherwise*

$\frac{vars(X)}{X}$

$\frac{vars(\_)}{mt}$

end

module *MProlog/Substitution*

imports *MProlog/Syntax*

imports *MProlog/VarAnalysis*

Fail :⟶ [*SubstitutionList*]

$\_,\_$: *SubstitutionList SubstitutionList* ⟶ *SubstitutionList* *associative identity*(*nilSL*)

$[\_\curvearrowright\_]$: *Var Term* ⟶ *Substitution*

apply : *TermList SubstitutionList* ⟶ *TermList*

apply2Subst : *SubstitutionList SubstitutionList* ⟶ *SubstitutionList*

idSubst : *VarSet* ⟶ *SubstitutionList*

nilSL :⟶ *SubstitutionList*

subst : *Var Term SubstitutionList* ⟶ *SubstitutionList*

subst : *Var Term TermList* ⟶ *TermList*

undefSL :⟶ [*SubstitutionList*]

$Fail \; , \frac{\_}{\cdot}$

$subst(X, T \_ \; , \frac{T" \; , \; Tl}{\cdot}) \; , \frac{\cdot}{subst(X, \; TT") \; , \; subst(X, \; TTl)}$

$\frac{\_}{\cdot} \; , \; Fail$

$\frac{\cdot}{[\; X \curvearrowright X \;]} \; , \; idSubst\langle \frac{X}{\cdot} \rangle$

$\frac{\cdot}{[\; X \curvearrowright apply(T, SL') \;]} \; , \; apply2Subst((\frac{[\; X \curvearrowright \; T \;]}{\cdot}\rangle, SL')$

$\frac{\cdot}{[\; Y \curvearrowright GT \;]} \; , \; subst(\_, \_(\frac{[\; Y \curvearrowright GT \;]}{\cdot}\rangle)$

$\frac{\cdot}{[\; Y \curvearrowright subst(X, TT') \;]} \; , \; subst(X, T(\frac{[\; Y \curvearrowright \; T' \;]}{\cdot}\rangle))$

$\dfrac{apply(\dfrac{Tl}{subst(X,\ T'Tl)},\ (\dfrac{[\ X \curvearrowright T'\,]\rangle)}{\cdot}}{\cdot}$

$\dfrac{apply(GTL,\ \_)}{GTL}$

$\dfrac{apply(Tl,\ nilSL)}{Tl}$

$\dfrac{apply2Subst(nilSL,\ \_)}{nilSL}$

$\dfrac{idSubst(mt)}{nilSL}$

$\dfrac{subst(X,\ TY)}{if\ \ X == Y\ then\ T\ else\ Y\ fi}$

$\dfrac{subst(X,\ T\ A\ (\ Tl\ ))}{A\ (\ subst(X,\ TTl)\ )}$

$\dfrac{subst(\_,\ \_GTL)}{GTL}$

$\dfrac{subst(\_,\ \_nilSL)}{nilSL}$

end

module *MProlog/Constraints*

imports *MProlog/Substitution*

imports *MProlog/Builtins*

$Clash :\rightarrow [ConstraintSet]$

$\_=\_ : TermList\ TermList \rightarrow Constraint$ *commutative*

$\_,\_ : ConstraintSet\ ConstraintSet \rightarrow ConstraintSet$ *associative commutative identity*($nilCS$)

$nilCS :\rightarrow ConstraintSet$

$subst : Var\ Term\ ConstraintSet \rightarrow ConstraintSet$

$\dfrac{Clash\ ,\ \dfrac{Clash}{\cdot}}{\cdot}$

$\dfrac{Clash\ ,\ \dfrac{\_}{\cdot}}{\cdot}$

$\dfrac{T\ ,\ \ T1\ ,\ Tl}{T1} = \dfrac{T'\ ,\ \ T1'\ ,\ Tl'}{T1'}\ ,\ \dfrac{\cdot}{T = T'\ ,\ \ Tl = Tl'}$

$\dfrac{subst(X,\ T\langle\ \dfrac{T' = T''}{\cdot}\rangle)\ ,\ \dfrac{\cdot}{subst(X,\ TT') = subst(X,\ TT'')}}{\cdot}$

$$\frac{. = (\_\rangle}{Clash}$$

$$\frac{GTL = GTL'}{if\ \ GTL == GTL'\ then\ nilCS\ else\ Clash\ fi}$$

$$\frac{Tl = Tl}{nilCS}$$

$$\frac{fail = \_}{Clash}$$

$$\frac{\_ = \_ (\_)}{Clash}$$

$$\frac{A\ (\ Tl\ ) = \ A'\ (\ Tl'\ )}{if\ \ A == A'\ then\ \ Tl = Tl'\ else\ Clash\ fi}$$

$$\frac{subst(\_, \_Clash)}{Clash}$$

$$\frac{subst(\_, \_nilCS)}{nilCS}$$

end

module *MProlog/ControlStack*
  imports *MProlog/Syntax*
  imports *MProlog/Substitution*
  imports *MProlog/K*
  $\_—\_ : StackElement\ Stack \rightarrow Stack$
  $(\_,\_,\_) : K\ RuleList\ SubstitutionList \rightarrow StackElement$
  $noStack :\rightarrow Stack$
end

module *MProlog/ResultList*
  imports *MProlog/Substitution*
  imports *MProlog/VarAnalysis*
  $No :\rightarrow ResultList$
  $Yes :\rightarrow ResultList$
  $\_;\_ : ResultList\ ResultList \rightarrow ResultList \qquad associative\ identity(nilRL)$
  $nilRL :\rightarrow ResultList$
end

module *MProlog/State*
  imports *MProlog/ControlStack*
  imports *MProlog/ResultList*
  $. :\rightarrow State$
  $\_\_ : State\ State \rightarrow State$ *associative commutative identity*(.)
  $answer : ResultList \rightarrow StateAttribute$
  $currRules : [RuleList] \rightarrow StateAttribute$
  $currSubst : SubstitutionList \rightarrow StateAttribute$
  $db : GroupedRulelistSet \rightarrow StateAttribute$
  $done :\rightarrow StateAttribute$
  $k : TermList \rightarrow StateAttribute$
  $nextVar : Nat \rightarrow StateAttribute$
  $numberOfSolutionsRequested : Int \rightarrow StateAttribute$
  $stack : Stack \rightarrow StateAttribute$
end

module *MProlog/Unification*
  imports *MProlog/Substitution*
  imports *MProlog/Constraints*
  imports *MProlog/VarAnalysis*
  $\{\_\} : ConstraintSet \rightarrow [SubstitutionList]$
  $unify : Term\ Term \rightarrow [SubstitutionList]$

$$\frac{\{\ Clash\ \}}{Fail}$$

$$\frac{\{\ nilCS\ \}}{nilSL}$$

$$\frac{\{\ \ Cs\ ,\ \ T = X\ \}}{if\ \ not\ \ X\ in\ vars(T)\ then\ \ \ [\ X \curvearrowright T\ ]\ ,\ \{\ subst(X, TCs)\ \}\ else\ Fail\ fi}$$

$$\frac{unify(T, T')}{\{\ \ T = T'\ \}}$$

end

module *MProlog/Search*
  imports *MProlog/State*
  imports *MProlog/Builtins*
  imports *MProlog/K*

imports *MProlog/Unification*
! : *Stack* → *BuiltInTerm*
V : *Int* → *Var*
applySubst : *TermList* → *Kitem*
applyToKitems : *SubstitutionList* → *Kitem*
freshVars : *VarSet* → *TermList*
rename : *TermList* → *Kitem*
renameSub : *SubstitutionList* → *Kitem*
renamed : *TermList* → *Kitem*
result : *State* → [*ResultList*]
storeStackInCuts : *TermList Stack* → *TermList*
sub : [*ResultList*] → *TermList*
unifyHeadWith : *Term* → *Kitem*
$\frac{currRules(\ ;\ )}{undef}\ k(\frac{\_}{fail})$
$\frac{currRules(undef)}{;}\ db(\_)\ k(\ \_\ (\ \_\ )\rangle \qquad otherwise$
$\frac{currRules(undef)}{Rl}\ db\langle\ \{\ A\ ,\ Rl\ \}\rangle\ k(\ A\ (\ \_\ )\rangle$
$\frac{currRules(\ \_\ )}{Rl'}\ currSubst(\frac{\_}{Sl'})\ k(\frac{fail \curvearrowright \_}{K'})\ stack(\frac{(\ K'\ ,\ Rl'\ ,\ Sl'\ )\ —\ St}{St})$
$\frac{currRules(\ Rf\ \text{:-}\ Rfl\ .\ Rl)}{undef}\ currSubst(Sl)\ k(\frac{T}{rename(\ Rf\ ,\ Rfl)\ \curvearrowright\ unifyHeadWith(T)}\ \curvearrowright K)\ stack(\frac{St}{(\ \ T \curvearrowright K\ ,\ Rl\ ,\ Sl\ )\ —\ St})$
$\frac{currSubst(\ Sl'\ )}{apply2Subst(Sl', Sl)}\ k(\frac{sub(Sl)}{applyToKitems(Sl)}\rangle$
$\frac{answer\langle\ \cdot\ )}{No}\ \frac{k(fail\rangle\ stack(noStack)}{done}$
$k(\ freshVars\langle\frac{X}{\cdot}\rangle \curvearrowright renameSub(\frac{\cdot}{[\ X \curvearrowright\ V(N)\ ]})\rangle\ nextVar(\frac{N\ }{s\ N})$
$k(\frac{!(St')}{\cdot}\rangle\ stack(\frac{\_}{St'})$
$k(\frac{renamed(\ T\ ,\ Tl) \curvearrowright unifyHeadWith(T')}{sub(unify(T, T')) \curvearrowright storeStackInCuts(Tl, St)}\rangle\ stack(\ \_\ —\ St)$
$\frac{\cdot}{if\ \ T == !\ then\ !(St)\ else\ T\ fi}\ ,\ storeStackInCuts((\frac{T}{\cdot}), St)$

$$\frac{answer(ResL)\ \ currSubst(Sl)\ \ k(.)\ numberOfSolutionsRequested(I)}{\begin{array}{l}\ if\ \ I ==\ \ s\ 0\ then\ \ done\ answer(\ ResL\ ;\ \ Sl\ ;\ Yes)\ else\ \ answer(\ ResL\ ;\ Sl)\ \ currSubst(Sl)\ \ k(fail)\ numberOfSolutionsRequested(\ I\ -\ \ s\ 0)\ fi \\ \underline{freshVars(mt)} \\ \qquad . \\ \qquad\qquad\qquad\qquad rename(\ Tl) \\ \hline freshVars(vars(\ Tl))\curvearrowright\ \ renameSub(nilSL)\curvearrowright\ applySubst(\ Tl) \\ \underline{storeStackInCuts(.,\_)} \\ \qquad\qquad . \\ \underline{sub(Fail)} \\ \quad fail \\ \underline{result\langle\ done\ answer(ResL)\rangle} \\ \qquad\quad ResL \\ (\ \underline{\qquad applyToKitems(Sl)\curvearrowright\ \ T\ ,\ Tl\qquad}\ \rangle \\ \quad apply(\ \ T\ ,\ Tl, Sl)\curvearrowright applyToKitems(Sl) \\ k\langle\underline{applyToKitems(\_)}) \\ \qquad\quad . \\ k(\ \underline{renameSub(Sl)\curvearrowright applySubst(\ Tl)}\rangle \\ \qquad renamed(apply(\ Tl, Sl)) \\ k(\ \underline{\qquad\qquad is\ (\ \ T\ ,\ T'\ )\qquad\qquad}\ \rangle \\ \quad sub(unify(intOrNat(eval(\ T')),\ T)) \\ k(\ \underline{\ \ T\ ,\ \ T'\ ,\ Tl\ \ }\ \rangle \\ \quad T\curvearrowright\ \ T'\curvearrowright Tl \\ k(\underline{grouped(\ Tl)}\rangle \\ \qquad Tl \end{array}}$$

end

module *MProlog/Interface*
  imports *MProlog/Search*
  imports *MProlog/PreprocessToGrouped*
  _;_? : *Database TermList* → *Query*
  _;_?_ : *Database TermList Nat* → *Query*
  [_] : *RuleList* → *Database*
  solve : *Query* → *ResultList*
  $$\frac{[\ Rl\ ]\ ;\ Fl\ ?}{[\ Rl\ ]\ ;\ Fl\ ?\ 0}$$

$$\frac{solve(\ [\ Rl\ ]\ ;\ Fl\ ?\ N)}{result(\ answer(nilRL)\ \ currRules(undef)\ \ currSubst(idSubst(vars(Fl)))\ \ db(\ prelude\ groupIt(Rl, noORL))\ \ k(Fl)\ \ nextVar(0)\ \ numberOfSolutionsRequested(N)\ stack(noStack))}$$

end